CAIM
Session 5: Page Rank

Octavi Pascual
Aleix Trasserra

November 9, 2016

1 Introduction

In this session we implemented the Page Rank algorithm. This algorithm is famous since it was designed and used by the founders of Google. Originally its goal was to measure the importance of website pages. However in this assignment we will see that the Page Rank idea can be used more generally in a directed graph. Our work is based on a network of airports and flights.

2 Implementation

We first implemented *data structures* for storing the routes and airports in a graph-based structure. In our implementation we have the following data structures:

- **Airport**: a class with the following fields:
  - **code**: IATA identifier of the airport.
  - **name**: complete name (city and country).
  - **routes**: set of airports that have this airport as destination.
  - **routeHash**: a map which each key belongs to the airports that have this airport as a destination and each value belongs to the weight of the edge.
  - **outdegree**: number of flights that have this airport as origin.
- **Edge**: a class with the following fields:
  - **origin**: code of the origin airport of the edge.
  - **weight**: number of pairs (origin,destiny).
- **edgelist**: list of edges.
- **edgeHash**: edgeHash[(origin, destination)] = weight.
- **airportList**: list of airports.
- **airportHash**: airportHash[code] = Airport.

All of the data structures above allows us to implement PageRank in a linear time. We implemented an iterative method based on the pseudo-code that was given to us.

As *stopping condition* we defined two conditions. The first condition is to put a maximum number of iterations that the algorithm can perform. For the second one, we implemented the following idea (False means that PageRank has not yet converged, true otherwise):

```
for all i in (P,Q):
   if (abs(Q[i] - P[i]) > epsilon) return False
end for
return True
```

One of the main issues is when a node has outweight equal to 0. We can't remove them because they have meaningful PageRank. To deal with it, our solution consists to assign the value $1/n$ to a node with outweight equal to 0. This solution works because it keeps the sum of Q after each iteration[1]. That is important because

3 Experimenting

Once we implemented the algorithm we can try to see how parameters affect it. First of all let's remember what were those parameters. The *number of maximum* iterations indicated how many iterations at most would the algorithm run. However, as we stated, it can end earlier if the page rank values have converged enough. In fact, if the difference between one iteration and another is lower than *epsilon* for each value, then we stop. Finally, the damping factor is the number used to stop other pages to having too much influence and to allow teleportation. The latter consists of adding a small probability to go from one page to another random page without following the edge restriction. This mimics when someone is surfing the web and decides to type another URL instead of following a hyperlink of the page where he actually is. After having explained the meaning of the parameters, let's study how their value affects the Page Rank algorithm.

- Max iterations
  This parameter is easy to explain. The longer it is, the more accurate results are. Increasing this number might not have any effect if the algorithm converges before the number of maximum iterations is reached.

- Epsilon
  This parameter is also responsible for the termination of the algorithm. The bigger epsilon is, the faster the computation will terminate.
  For example, let's run the program with the following fixed parameters: MAX_ITERATIONS = 100, DAMPING_FACTOR = 0.85. We modify EPSILON:

  - EPSILON = 0.1
    It took only one iteration for results to converge. We can easily explain that since at the first iteration we set the page rank to 1/n to all the airports. This value is way lower than 0.1.
  - EPSILON = 0.00001
    It took 22 iterations for the page rank values to converge.
  - EPSILON = 0.0000000000000000000000001
    Here we set epsilon to a really low value. It takes 100 iterations, the maximum number of iterations, since with such a small value it takes longer to converge.

- Damping Factor
  We can see that the higher the damping factor is, the longest it takes for the algorithm to converge. Aversely, the lowest it is the less iterations are required but resulsts are less accurate. It is a trade-off between time and precision.
  For example, we run the program with the following fixed parameters: EPSILON = 0.000001, MAX_ITERATIONS = 1000. Now we try to modify DAMPING_FACTOR to see what happens.
  - DAMPING_FACTOR = 0.20
    It took 5 iterations for the page rank values to converge.
  - DAMPING_FACTOR = 0.70
    It took 17 iterations for the page rank values to converge.
  - DAMPING_FACTOR = 0.85
    It took 37 iterations for the page rank values to converge.
  - DAMPING_FACTOR = 0.99
    It took 583 iterations for the page rank values to converge.

4 Conclusion

In conclusion we can say that Page Rank is an algorithm that looks simple but what it is surprising is the number of different ways and techniques that exist to compute it. It is interesting to think about how it can be implemented in the World Wide Web. There, the network is composed by millions of nodes and millions of edges and probably this algorithm would be too slow. Also, the network itself would be too big to fit in memory during the execution of this algorithm. It is challenging to design a method to be able to apply Page Rank to such a big network.

References

[1] Deeper Inside PageRank - The University of Chicago
`http://galton.uchicago.edu/~lekheng/meetings/mathofranking/ref/langville.pdf`